View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by Elsevier - Publisher Connector

Comp. & Maths. with Appls., Vol. 4, pp. 15–31. Pergamon Press, 1978. Printed in Great Britain

# AN IMPLICIT ENUMERATION ALGORITHM FOR THE ALL INTEGER PROGRAMMING PROBLEM

ANIL B. JAMBEKAR
Michigan Technological University, Houghton, Michigan 49931, U.S.A.

and

DAVID I. STEINBERG
Southern Illinois University, Edwardsville, Illinois 62026, U.S.A.

(*Received June* 1977)

**Abstract**—In this paper an implicit enumeration algorithm is presented for solving the general integer programming problem. The algorithm is a generalization of algorithms of Balas and Geoffrion for solving 0–1 problems. In order that each variable may be treated directly, a special data structure has been designed. Based on this data structure, a procedure is developed which efficiently keeps track of the status of the algorithm and of the potential actions to be taken at any stage in the procedure. The concept of a 'best' surrogate constraint is extended and incorporated into the algorithm. The algorithm can be modified for solving mixed integer programming problems.

## 1. INTRODUCTION

The general all integer programming problem can be written as:

Maximize the linear form

$$z = cx$$

subject to $Ax \leq b,$

$$x \geq 0, \quad x \text{ integral (i.e. all components of } x \text{ are integers)} \tag{1}$$

where $x$ is a $n$-vector to be determined,
$c$ is a given $n$-vector of costs,
$b$ is a given $n$-vector of costs,
$b$ is a given $m$-vector of requirements,
$A$ is a given $m \times n$ matrix.

The algorithm for an all integer linear programming problem presented here can be considered as a generalization of the approach to the zero–one problem by Geoffrion[1]. The variables are treated directly as integer variables avoiding the computationally costly artifice of mapping each variable to two or more zero–one variables through binary representation. This leads to a problem of designing the appropriate data structure for efficient computerized book-keeping to take a maximum advantage of the full integer range available to each variable. Here a procedure is developed, which can effectively keep track of the status of the algorithm and the potential actions to be taken at any point. In addition the concept of a surrogate constraint is extended theoretically. Other implicit enumeration-type algorithms include those of Balas[2, 3], Geoffrion[1, 4], Glover[5], Krolak[6, 7], Lemke and Spielberg[8], and Land and Doig[9].

The decomposition principle is used in the linear programming part of the algorithm in order to simplify post optimality calculations. It also allows the linear programming iterations to be started from any known integer solution, not necessarily an extreme point of the convex set formed by the constraints. Also a heuristic approach is suggested, which can be imbedded in the algorithm to improve computational efficiency.

Although we shall describe the algorithm in terms of the all integer programming problem, it

can be modified in a straightforward manner to handle mixed integer programming problems[10]. It is assumed here that $c$ has all non-negative integer components and that bounds on $x$ are known or can be estimated. Denote $l$ and $u$ as $n$-vectors of lower and upper bounds on $x$, respectively. Let $I$ be the set of all integer points, i.e. $I = \{x | l \leq x \leq u, x \text{ integral}\}$.

The algorithm to be considered in this paper can be best described in terms of a search procedure. A complete search accounts in some way for each of the integer points in the set $I$. The basic method is a skeleton method of pursuing the complete search, but embodying techniques which exclude large numbers of solutions from further consideration. The search algorithm consists of resolving the problem in $n$ variables, i.e. all possible solutions have to be enumerated explicitly or implicitly. Resolving a problem in $k$ variables means resolving all problems in $(k-1)$ variables which result from fixing one variable to all permissible integer values. To effect the search in $k$ variables,

(1) fix some one variable, $x_j$, to one of its permissible integer values,

(2) resolve the problem in the remaining variables,

(3) fix $x_j$ to another permissible value,

(4) repeat steps (2) and (3) until all possible values for $x_j$ are considered. This basic search implies a general state of search in which all possible solutions are considered either explicitly or implicitly. This plan is shown schematically in Fig. 1.

Explicit generation of all integer solutions for large values of $k$ is computationally prohibitive. A clever manipulation of blocks 2, 3 and 5 will make this process practical, which in turn will produce an exact algorithm to solve problem (1).

In the next section the basic tree structure upon with the algorithm depends is described. Also described in later sections are the various techniques used to exclude a large number of solutions from further consideration. Also described is a heuristic which can be employed in order to improve the computational efficiency of the algorithm. All of these are concerned with the manipulation of blocks 2, 3 and 5 in the process specified in Fig. 1.

## 2. THE BASIC TREE STRUCTURE AND TERMINOLOGY

In this section of a basic framework and terminology will be developed for understanding the algorithm. The procedure consists of making a choice of a variable to fix, followed by a choice of an integer value at which to fix it, and determining what restrictions will be imposed

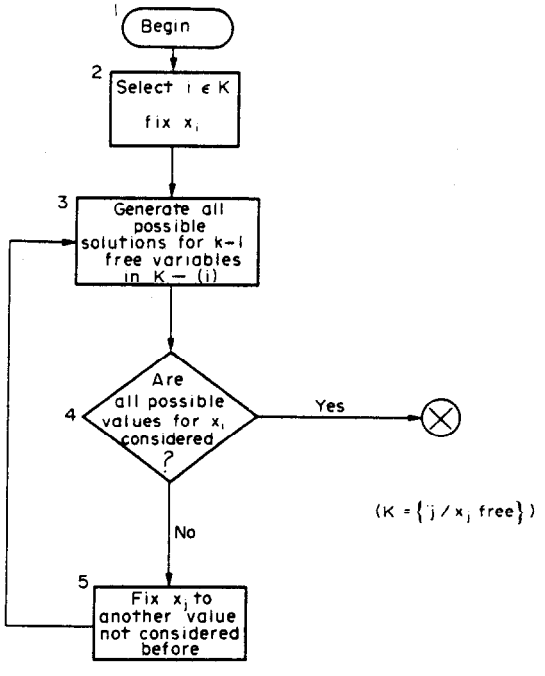


Fig. 1. A search in $k$ variables.

on the remaining variables. As the algorithm proceeds, a variable is chosen and fixed to some specific value. In other words, this variable is fixed explicitly to a certain value which is chosen by a predetermined rule. For a given set of fixed variables the free variables are allowed to take only specific values in order to assure feasibility. In this contest a free variable is said to be fixed *implicitly* if it can take only one value. For other free variables, the values that they can take are restricted by certain bounds, which are termed 'conditional bounds for a given set of fixed variables'. For a variable said to be implicitly fixed, this implies that the conditional upper bound is equal to the conditional lower bound.

To be able to identify the current status of the algorithm, define the term *state*, which indicates the level of the tree structure at which the algorithm is currently operating. Let a state variable $s$ denote the current level of the tree structure. A new state is said to be reached every time a variable is fixed explicitly.

In terms of the tree structure, Fig. 2 depicts the current state $s$, which is reached by a variable $x_{j_s}$ at state $(s-1)$. Currently $s$ variables are explicitly fixed; $k$ variables are implicitly fixed in state 0 to $(s-1)$ and the problem to resolve is in the $f_s = n-(s+k)$ free variables.

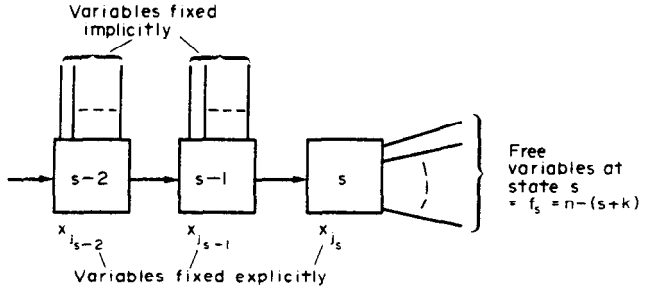


Fig. 2. A tree search.

After fixing the variable $x_{j_s}$, the conditional bounds on the remaining free variables must be determined. Let $l_j^{s+1}$ and $u_j^{s+1}$ be the conditional lower and upper bounds determined for any free variable $x_j$ at the current state $s$. Any free variable $x_j$ is said to be fixed implicitly, if $l_j^{s+1}$ is equal to $u_j^{s+1}$. If $l_j^{s+1}$ is greater than $u_j^{s+1}$ for any free variable $x_j$, then the solution set is empty for the current value of $x_{j_s}$. In this case a new assignment to $x_{j_s}$ must be made. Once all the problems in the $f_s$ free variables resulting from every *possible* assignment to $x_{j_s}$ are resolved. the problem in $f_{s-1}$ free variables at state $(s-1)$ resulting from the assignment to $x_{j_{s-1}}$ is said to be resolved.

This procedure requires a definite sequencing of all the points in $I = \{x | l \le x \le u\}$. Since this sequencing is entirely dependent upon the particular problem being solved, a certain way must be devised to record the status of the current and previous states and potential actions that may be taken at each state. To accomplish this efficiently a $5 \times n$ matrix $\eta$, is maintained, and this matrix is updated as necessary during the process. The first row of $\eta$ corresponds to the variable numbers. At any state $s$, the first $(s+k)$ columns will have entries. The variables which are fixed explicitly or implicitly are recorded left to right in the order in which they are fixed. Thus at state $s$, entry in the $(s+k)$th position in the first row, is $j_s$. In the process of resolving the problem in $f_s = n-(s+k)$ variables, all the variables fixed implicitly, are recorded in the succeeding positions. The second row of $\eta$ contains the values at which the variables in the first row are fixed. It is necessary to know what the next value should be assigned to the variable which is fixed explicitly, when the problem resulting from the current assignment is resolved. This value is entered in the third row of $\eta$. The fourth and fifth rows are used to record the conditional bounds on the fixed variables at the instance they are fixed. This will help in identifying the implicitly fixed variables from the explicitly fixed variables, and is simply a book-keeping procedure.

## 3. CONDITIONAL BOUNDS AND SOME RELATED EXISTENCE THEOREMS

An assignment of permissible integer values to a subset of $n$ variables in $x$ is called a partial solution. Any variable not assigned a value is termed free. Note that a variable can be assigned

a value explicitly or implicitly. The bounds on the free variables are called the *conditional bounds*. At a given state $s$, let $\hat{J}_s$ denote the set of fixed variables, and let $J_s = \{j | x_j \in \hat{J}_s\}$. Let $N = \{1, 2, \ldots, n\}$. Let $l_j^{s+1}$ and $u_j^{s+1}$ be the conditional bounds to be determined for a free variable $x_j$ for a given set of fixed solution to (2) and state $s$. Adding the constraint $cx \geq z_l$ to the original set of constraints, yields the following problem to be resolved:

Maximize

$$z^s = \sum_{j \in N - J_s} c_j x_j$$

subject to

$$\sum_{j \in N - J_s} a_{ij} x_j \leq b_i - \sum_{j \in J_s} a_{ij} x_j, \; i = 1, \ldots, m + k, \qquad x_j \geq 0 \text{ and integer}, \; j \in N - J_s \tag{2}$$

where the $(m+1)^{st}$ constraint is $cx \geq z_l$, and the last $(k-1)$ constraints are the so called 'surrogate constraints' to be defined later.

At this point, the lower and upper bounds on any $x_j$, $j \in N - J_s$, are given by $l_j^s$ and $u_j^s$, which were determined at state $(s-1)$. From these bounds and the constraint set in (2), new conditional bounds $l_j^{s+1}$ and $u_j^{s+1}$ are determined. Note that the problem (2) formulated at state $s$ is more restrictive than that formulated at the previous state $(s-1)$, i.e. there is an additional restriction imposed by fixing $x_{j_s}$ to reach state $s$. This implies the following:

$$l_j^{s+1} \geq j^s \qquad \text{and} \qquad u_j^{s+1} \leq u_j^s.$$

Now the calculation of the conditional bounds for state $s$ will be considered. There are various ways to determine these bounds. One way is by solving two linear programming problems for each free variable, i.e. maximizing and minimizing each free variable over the set of constraints given in problem (2). However, this approach is computationally expensive, and hence, an approach based on simple observation is much more attractive, even if the bounds obtained are not as good. With this in mind, consider

$$s_i^s = b_i - \sum_{j \in J_s} a_{ij} x_j - \sum_{\substack{j \in N - J_s \\ a_{ij} > 0}} a_{ij} l_j^s - \sum_{\substack{j \in N - J_s \\ a_{ij} < 0}} a_{ij} u_j^s \tag{3}$$

which can be written as

$$s_i^s = b_i^s - \sum_{\substack{j \in N - J_s \\ a_{ij} > 0}} a_{ij} l_j^s - \sum_{\substack{j \in N - J_s \\ a_{ij} < 0}} a_{ij} u_j^s, \qquad i = 1, \ldots, m + 1. \tag{4}$$

where

$$b_i^s = b_i - \sum_{j \in J_s} a_{ij} x_j.$$

Now, consider each constraint separately; consider only those free variables for which $a_{ij} \neq 0$. There are two cases to consider.

*Case* 1. $(a_{ij} > 0)$. The $i^{th}$ constraint can be rewritten as

$$a_{ij} x_j \leq b_i^s - \sum_{\substack{k \in N - J_s \\ k \neq j}} a_{ik} x_k. \tag{5}$$

Noting that $l_k^s \leq x_k \leq u_k^s$, gives

$$a_{ij} x_j \leq b_i^s - \sum_{\substack{k \in N - J_S \\ k \neq j \\ a_{ik} > 0}} a_{ik} l_k^s - \sum_{\substack{k \in N - J_s \\ a_{ik} < 0}} a_{ik} u_k^s \tag{6}$$

and from (4) and (6)

$$a_{ij}x_j \leq s_i^s + a_{ij}l_j^s + a_{ij}l_j^s. \tag{7}$$

Now it follows that

$$x_j \leq l_i^s + \left\lfloor \frac{s_i^s}{a_{ij}} \right\rfloor \tag{8}$$

where $\lfloor Q \rfloor$ denotes the largest integer not greater than $Q$. It can be seen that $x_j$ can be no larger than

$$u_j^{s+1} = \min \left\{ u_i^s, \min_{\substack{\text{all } i \\ a_{ij}>0}} \left\{ l_j^s + \left\lfloor \frac{s_i^s}{a_{ij}} \right\rfloor \right\} \right\} \tag{9}$$

*Case* 2. ($a_{ij} < 0$). The argument here is similar to Case 1. The final result is

$$l_j^{s+1} = \max \left\{ l_j^s, \max_{\substack{\text{all } i \\ a_{ij}<0}} \left\{ u_j^s - \left\lceil \frac{s_i^s}{a_{ij}} \right\rceil \right\} \right\} \tag{10}$$

or

$$l_j^{s+1} = \max \left\{ l_j^s, \max_{\substack{\text{all } i \\ a_{ij}<0}} \left\{ u_j^s + \left\lceil \frac{s_i^s}{a_{ij}} \right\rceil \right\} \right\}$$

where $\lceil Q \rceil$ denotes the smallest integer not less than $Q$.

It can easily be verified from (9) and (10), that if $s_i^s$ is less than zero, the solution set for problem (2) is empty. Since $s_i^s < 0$ implies that $u_j^{s+1} < l_j^s$ for $a_{ij} > 0$, or that $l_j^{s+1} > u_j^s$ for $a_{ij} < 0$, no solution to problem (2) exists. Note that $u_j^{s+1} = l_j^{s+1}$ implies $x_j$ is implicitly fixed. Thus, expressions (9) and (10) give an iteration scheme to determine conditional bounds at state $s$. They provide a necessary, but not sufficient condition, for the existence of an integer solution for problem (2).

Now consider a solution $\bar{x}$ formed by the fixed variables and by setting the free variables at their upper bounds determined as above. For this solution let us define a slack variable $y_i^s$, for constraint $i$.

$$y_i^s = b_i^s - \sum_{j \in N - J_s} a_{ij}u_j^{s+1}, \qquad i = 1, \ldots, m+1. \tag{11}$$

The corresponding value of the objective function at state $s$ is

$$z^s = c\bar{x} = \sum_{j \in J_s} c_j x_j + \sum_{j \in N - J_s} c_j u_j^{s+1} \tag{12}$$

It is obvious that if all $y_i^s$ are greater than or equal to zero, we have an improved solution. For the case that some $y_i^s$ are less than zero consider the following theorem:

*Theorem* 1. For any $y_i^s$ less than zero, if

$$\sum_{\substack{j \in N - J_s \\ a_{ij}>0}} a_{ij}(u_j^{s+1} - j^{s+1}) < |y_i^s|,$$

no feasible integer solution to problem (2) exists.

*Proof.* Any constraint $i$ in problem (2), for which $y_i^s$ is less than zero, can be written as:

$$\sum_{\substack{j \in N-J_s \\ a_{ij}>0}} a_{ij}x_j + \sum_{\substack{j \in N-J_s \\ a_{ij}<0}} a_{ij}x_j \leq b_i^s. \tag{13}$$

Minimizing the left hand side of (13) over the set $\{l_j^{s+1} \leq x_j \leq u_j^{s+1}, j \in N - J_s\}$ gives

$$\sum_{\substack{j \in N-J_s \\ a_{ij}>0}} a_{ij}l_j^{s+1} + \sum_{\substack{j \in N-J_s \\ a_{ij}<0}} a_{ij}u_j^{s+1}. \tag{14}$$

A necessary condition for the existence of a feasible solution to (13) is that

$$\sum_{\substack{j \in N-J_s \\ a_{ij}>0}} a_{ij}j^{s+1} + \sum_{\substack{j \in N-J_s \\ a_{ij}<0}} a_{ij}u_j^{s+1} \leq b_i^s. \tag{15}$$

Writing expression (11) as

$$y_i^s = b_i^s - \sum_{\substack{j \in N-J_s \\ a_{ij}>0}} a_{ij}u_j^{s+1} - \sum_{\substack{j \in N-J_s \\ a_{ij}<0}} a_{ij}u_j^{s+1}$$

or

$$y_i^s + b_i^s - \sum_{\substack{j \in N-J_s \\ a_{ij}>0}} a_{ij}(u_j^{s+1} - l_j^{s+1}) - \sum_{\substack{j \in N-J_s \\ a_{ij}<0}} a_{ij}u_j^{s+1} - \sum_{\substack{j \in N-J_s \\ a_{ij}>0}} a_{ij}l_j^{s+1} \tag{16}$$

gives

$$\sum_{\substack{j \in N-J_s \\ a_{ij}>0}} a_{ij}(u_j^{s+1} - l_j^{s+1}) = b_i^s - \sum_{\substack{j \in N-J_s \\ a_{ij}<0}} a_{ij}u_j^{s+1} - \sum_{\substack{j \in N-J_s \\ a_{ij}>0}} a_{ij}l_j^{s+1} - y_i^s. \tag{17}$$

Applying (15) yields an equivalent necessary condition,

$$\sum_{\substack{j \in N-J_s \\ a_{ij}>0}} a_{ij}(u_j^{s+1} - l_j^{s+1}) \geq -y_i^s. \tag{18}$$

This implies if

$$\sum_{\substack{j \in N-J_s \\ a_{ij}>0}} a_{ij}(u_j^{s+1} - l_j^{s+1}) < |y_i^s|, y_i^s < 0$$

problem (2) does not possess a solution for the current partial solution at state $s$.

Another theorem relating to the existence of solutions can be stated as follows;

*Theorem* 2. If $z_l$ is the best known solution at state $s$, then for any $i^{\text{th}}$ constraint for which $y_i^s < 0$, if

$$\frac{c_j}{a_{ij}} > \frac{z^s - z_l}{|y_i^s|} \tag{19}$$

for all $a_{ij} > 0$ and $j \in N - J_s$, no better feasible solution to problem (2) exists for the partial solution at state $s$.

*Proof*: This theorem is proved by contradiction and by application of Theorem 2.

Assume (19) is true and that there exists a solution $x^*$ such that $z_l \leq c \cdot x^* \leq z^s$. For state $s$, the components of $x^*$ corresponding to the free variables, i.e., $j \in N - J_s$, lie between $l_j^{s+1}$ and $u_j^{s+1}$.

If $x^*$ is a feasible solution, then by Theorem 1 it must be true that

$$\sum_{\substack{j \in N-J_s \\ a_{ij}>0}} a_{ij}(u_j^{s+1} - x_j) > |y_i^s| \qquad \text{for } y_i^s < 0. \tag{20}$$

Multiplying and dividing each term on the left by $c_j$ yields

$$\sum_{\substack{j \in N-J_s \\ a_{ij}>0}} \frac{a_{ij}c_j(u_j^{s+1} - x_j^*)}{c_j} > |y_i^s|$$

and from (19) it follows that

$$\frac{|y_i^s|}{z^s - z_l} \sum_{\substack{j \in N-J_s \\ a_{ij}>0}} c_j(u_j^{s+1} - x_j^*) > |y_i^s|. \tag{21}$$

Note that

$$z_l < \sum_{j=1}^{n} c_j x_j^* < z^s$$

which implies

$$z^s - \sum_{j=1}^{n} c_j x_j^* < z^s - z_l$$

or

$$\sum_{j=1}^{n} c_j(u_j^{s+1} - x_j^*) < z^s - z_l$$

or

$$\sum_{\substack{j \in N-J_s \\ a_{ij}>0}} c_j(u_j^{s+1} - x_j^*) < z^s - z_l. \tag{22}$$

Upon combining (22) with (21), it can be seen that (21) is a contradiction.

The procedure for determining the conditional bounds and the results of Theorems 1 and 2 provide what appears to be some fairly powerful tools to use in an integer linear programming algorithm.

## 4. THE LINEAR PROGRAM AND ITS USE IN THE PROPOSED ALGORITHM

Linear programming plays an important role in this algorithm. One of its uses was mentioned in the last section; much tighter conditional bounds on each of the free variables can be obtained, if a linear programming method is used. Once the conditional bounds are determined at state $s$, the resulting problem is:

Maximize

$$z^s = \sum_{j \in N-J_s} c_j x_j$$

subject to

$$\sum_{j \in N-J_s} a_{ij}x_j \leq b_i^s, \quad i = 1, \ldots, m+1.$$

$$l_j^{s+1} \leq x_j \leq u_j^{s+1}, \quad j \in N - J_s \qquad x_j \text{ integer}, j \in N - J_s. \tag{23}$$

If problem (23) is solved as a continuous linear program, three outcomes are possible:

(1) no feasible solution exists,

(2) an optimal feasible solution exists, which satisfies the integrality requirement,

(3) an optimal feasible solution exists, but no optimal feasible solution satisfies the integrality requirement.

In the first instance, no feasible integer solution can be obtained for the current partial solution at state $s$, which causes the algorithm to revert back to state $(s-1)$, and a new assignment to $x_{j_s}$ is made.

In the second instance, if the optimal solution satisfies the integrality requirement, an improved integer solution is obtained and the procedure again reverts to state $(s-1)$.

In the first and second instances, one can also conclude the following. If $x^*_{j_s}$ is the exact fractional value of the variable $x_{j_s}$ in the continuous linear program solved at state $(s-1)$, no integer feasible solution can be achieved for the partial solution formed at state $s$ by assigning $x_{j_s}$ any value less (greater) than $x^*_{j_s}$, if the current assignment of $x_{j_s}$ is less (greater) than $\bar{x}_{j_s}$. This follows from a well known theorem of Land and Doig[9]:

"If $z(x_j)$ is the value of the objective function at optimality for the various values of $x_j$, then $z(x_j)$ is a concave function of $x_j$."

In the third instance, no conclusive decisions can be made at state $s$. The procedure advances to state $(s+1)$. It can now be seen what is to be done in this case. First, the linear programming solution gives a local upper bound on the integer solutions for any state following state $s$. Also, this solution can be used to generate a redundant, but very useful constraint, called the *surrogate constraint*. This new constraint tends to improve the conditional bounds, and thus helps to restrict the total search.

Geoffion's definition of a surrogate constraint can be extended to this problem. Let $v$ be a $m$-component non-negative vector. A surrogate constraint can be written as

$$v(b - Ax) + cx - z_l > 0, \qquad v \geq 0 \tag{24}$$

for a given $v$. A surrogate constraint $v_1(b-Ax)+cx-z_l>0$ is said to be stronger than the surrogate constraint $v_2(b-Ax)+cx-z_l>0$ relative to the partial assignment at state $s$, if

$$\max_{x_j,\, j \in N-J_s} [v_1(b-Ax)+cx-z_l) < \max_{x_j,\, j \in N-J_s} [v_2(b-Ax)+cx-z_l] \tag{25}$$

In order to find the strongest surrogate constraint, in the sense of the above definition, solve

$$\min_{v \geq 0} \{ \max_{j \in N-J_s} [v(b-Ax)+cx-z_l] \} = \min_{v \geq 0} \left\{ \sum_{i=1}^{m} v_i b_i + \sum_{j \in J_s} \left( c_j - \sum_{i=i}^{m} v_i a_{ij} \right) x_i - z_l \right.$$

$$\left. + \max \left[ \sum_{j \in N-J_s} \left( c_j - \sum_{i=1}^{m} v_i a_{ij} \right) x_j \middle| l_j^{s+1} \leq x_j \leq u_j^{s+1};\ x_j \text{ integer},\ i \in N - J_s \right] \right\}. \tag{26}$$

We can write the term to be maximized in (26) as

$$\max \left[ \sum_{j \in N-J_s} \left( c_j - \sum_{i=1}^{m} v_i a_{ij} \right) x_j \leq u_j^{s+1},\ -x_j \leq -l_j^{s+1},\ x_j \geq 0 \right]. \tag{27}$$

$$= \min \left[ \sum_{j \in N-J_s} u_j^{s+1} w^{ju} - \sum_{j \in N-J_s} l_j^{s+1} w_j^l \middle| w_j^u, w_j^l \geq 0 \text{ and } w_j^u - w_j^l \geq c_j - \sum_{i=1}^{m} v_i a_{ij}, \text{ for } j \in N - J_s \right], \tag{28}$$

where $w_j^u$, $w_j^l$ are dual variables.

Substituting (28) into (26) yields the following linear program

$$\min_{v,\, w_j^u,\, w_j^l} \left\{ \sum_{j \in J_s} c_j x_j - z_l + \sum_{i=1}^{m} v_i \left( b_i - \sum_{j \in J_s} a_{ij} x_j \right) + \sum_{j \in N - J_s} (u_j^{s+1} w_j^u - l_j^{s+1} w_j^l) \right\} \tag{29}$$

subject to

$$v_i,\ w_j^u,\ w_j^l > 0, \text{ all } i \text{ and } j,\ w_j^u - w_j^l \geq c_j - \sum_{i=1}^{m} v_i a_{ij},\ j \in N - J_s.$$

The solution to (29) provides the strongest surrogate constraint.

Note that the dual problem to (29) is the problem (23), as a continuous linear program. Hence (29) need not be solved explicitly, as the necessary information can be derived from the optimal solution to (23) solved as a continuous linear program.

Consider the computational aspects of problem (23) as a linear program. From state to state, the bounds on the variables vary, sometimes drastically. This might make the solution to the linear program obtained by the ordinary primal simplex method, very expensive, because the variation in the lower and/or upper bound constraints either involve many extra calculations or necessitate re-solving the problem. To avoid these apparent difficulties, the principle of decomposition is used, and this also appears to minimize both the amount of computation and storage requirements on a computer. For a more detailed explanation of the use of the principle of decomposition in this manner see[10]. In the next section we will see how this method of solving the linear program can facilitate the 'heuristic search' which in turn reduces the overall computation.

One should bear in mind that the proper choice of which free variable to fix, and the proper value at which to fix it, can greatly affect the efficiency of the algorithm. It will be seen how the linear programming solution can help in making these choices in the section on 'choice of a variable to fix'.

## 5. A HEURISTIC SEARCH

A heuristic search can be imbedded in the algorithm to improve the overall computational efficiency. One might suspect that the efficiency of the algorithm would be sensitive to the initial known solution. By making frequent use of the proposed heuristic search initially a good solution can be obtained, which can considerably reduce the total amount of computation necessary.

The basic idea is very simple. Pick two points, $x_1$ and $x_2$, in such a way that the line segment joining these two points is most likely to pass through the feasible region. Move along this line and make a search in the neighborhood of the points on the line. Once a solution is found, a simple direct search can be employed to obtain a better solution. The direct search involves changing one variable at a time, while still maintaining the feasilibity and thus obtaining a locally optimal solution in the sense of this search.

Since the two points selected are arbitrary, one has a complete freedom of choosing these two points. Prior knowledge of the structure of the problem being solved can influence the choice of the points. Some of the ways of picking $x_1$ and $x_2$ are suggested here. For a given state $s$, $x_1$ can be chosen as a linear programming solution at state $s$ and $x_2$ as best known integer solution. The line segment joining these two points certainly passes through the feasible region. Another way to select $x_1$ and $x_2$ is as follows. At given state $s$, $x_1$ can be formed by the variables in $J_s$ and by setting the free variables to their respective conditional upper bounds; similarly $x_2$ can be formed by the variables in $J_s$ and by setting the free variables to their respective conditional lower bounds.

This search can be used more advantageously in the initial period of computation, e.g. while solving a linear program the first time. Since the linear program is solved using the principle of decomposition, at every iteration an extreme point of the set $\{l' \leq x \leq u'\}$ is obtained; $x_1$ chosen as this generated extreme point and $x_2$ as the extreme point which lies diagonally opposite to $x_1$ in the set $\{l' \leq x \leq u'\}$. For example, if $l' = (3,0,1)$ and $u' = (6,4,3)$ and the generated point is (3,4,1), $x_1$ and $x_2$ are (3,4,1) and (6,0,3) respectively. In this manner one would expect to find a

good integer solution at the same time the solution to the first linear program is obtained. Note that a heuristic algorithm for solving an all integer problem can be obtained by solving the associated linear program by the principle of decomposition and by employing the embedded search described here.

Let $z_1$ and $z_2$ be the values of the objective function at $x_1$ and $x_2$ respectively. Without any loss of generality, it can be assumed that $z_1 > z_2$. Let $z_u$ and $z_l$ be the current upper and lower bounds on the integer solutions. Any point $x$ on the line segment joining $x_1$ and $x_2$ can be written as

$$x = \lambda x_1 + (1-\lambda)x_2, \qquad 0 \le \lambda \le 1. \tag{30}$$

Multiplying both sides of equation (30) by the vector $c$ gives

$$cx = \lambda cx_1 + (1-\lambda)cx_2$$

or

$$z = \lambda z_1 + (1-\lambda)z_2 \tag{31}$$

Expression (31) is further used to estimate the range of $\lambda$. Note that only those values of $z$ need to be considered which satisfy

$$z_l \le z \le z_u.$$

Substituting for $z$ and simplifying give

$$\frac{z_l - z_2}{z_1 - z_2} \le \lambda \le \frac{z_u - z_2}{z_1 - z_2} \tag{32}$$

It follows from (30) and (32) that

$$\max\left[0, \frac{z_l - z_2}{z_1 - z_2}\right] \le \lambda \le \min\left[1, \frac{z_u - z_2}{z_1 - z_2}\right]. \tag{33}$$

Now, for various values of $\lambda$ in the range defined by (33), calculate $x$ from (30) and then check the feasibility of the following three points;

(1) the point obtained by truncating each component of $x$,

(2) by bounding each component of $x$, and

(3) by truncating each component of $(x+1)$, where '$1$' here is a $n$-component vector whose components are each unity.

Note that these points are arbitrarily chosen. If this procedure gives a feasible integer solution, use this solution as an initial point for further direct search. During the direct search, an attempt is made to obtain a better solution by varying one variable at a time while still maintaining feasibility. Let $x^*$ be the feasible integer solution obtained by the above procedure. The slack variables for the $x^*$ are greater than or equal to zero and are given by

$$sv_i = b_i - \sum_{j=1}^{n} a_{ij}x_j^*, \quad i = 1, \ldots, m. \tag{34}$$

Next, define an integral value increment for each variable $x_j$:

$$\Delta x_j = \min_{\substack{\text{all } i \\ a_{ij}>0}} \left\{ \left\lfloor \frac{sv_i}{2a_{ij}} \right\rfloor \right\} \tag{35}$$

where the constant 2 has been chosen arbitrarily. This is similar to the procedure used by Echols[12] in his direct search method. The idea is not to get too close to the boundaries.

Select $x_k$ for which the potential contribution $c_k . \Delta x_k$ to the objective function is maximum. Move $x^*$ in the $k$th direction by $\Delta x_k$. Repeat the procedure until no improvement can be made. Then double the increment for each variable $x_j$ and the repeat the above procedure.

The whole heuristic procedure can be summarized as follows:

Given $x_1$, $x_2$ and corresponding values $z_1$, $z_2$. Also known $z_l$ and $z_u$.

1. Find the range for $\lambda$ using (33) and a select a step size $\lambda\Delta$. Set $\Delta$ to its upper bound.
2. Find $x = \lambda x_1 + (1-\lambda)x_2$.
3. Check the feasibility of the following points obtained by

   a. truncating $(x+1)$
   b. truncating $(x+0{\cdot}5)$, and
   c. truncating $x$.

If a feasible solution is obtained, go to step 5.

4. Set $\lambda = \lambda - \Delta\lambda$.

If $\lambda$ is lower than its lower bound, terminate the procedure, otherwise go to step 2.

5. Set $x^*$ equal to the feasible solution obtained in step 3. Set $d = 2$, where $d$ is an arbitrary constant.
6. Calculate $sv_i = b_i - \sum_{j=1}^{n} a_{ij}x_j^*$ for $i = 1, \ldots, m$).
7. Select $x_k$ which maximizes the potential contribution, i.e., $c_k\Delta x_k = \max_{\text{all } j} \{c_j \Delta x_j\}$ where

$$\Delta x_j = \min_{\substack{\text{all } i \\ a_{ij}>0}} \left\{ \left\lfloor \frac{sv_i}{da_j} \right\rfloor \right\}.$$

8. If $c_k . \Delta x_k = 0$ and $d = 2$, set $d = 1$ and go to step 7. If $c_k . \Delta x_k = 0$ and $d = 1$, terminate the procedure.
9. Make a move from $(x_1^*, x_2^*, \ldots, x_k^*, \ldots, x_n^*)$ to $(x_1^*, x_2^*, \ldots, x_k^* + \Delta x_k, \ldots, x_n^*)$ and go to step 6.

This procedure appears to be rapid and relatively effective. As it is based completely on intuition, one has complete flexibility in using it or making any changes.

## 6. CHOICE OF A VARIABLE TO FIX

The efficiency of the algorithm can depend upon the choice of a free variable to fix and the value at which to fix it. For any given state $s$ on every backward step the variable $x_{j_s}$ is assigned another value which has not been previously considered. For an efficient algorithm it is essential to order these values in such a way that the total number of values that will be considered is minimized.

Consider first the choice of which free variable to fix at state $s$. If the solution to the resulting linear program at state $(s-1)$ is feasible and at optimality it does not satisfy the integrality requirement, the algorithm advances to state $s$ by fixing a free variable $x_{j_s}$. Consider only those free variables which do not assume integrality in the optimal solution to the linear program. The use of Theorem 2 is made to determine a value criterion for $x_{j_s}$. Theorem 2 can be stated for state $(s-1)$ as follows:

If for all $a_{ij} > 0$, $j \in N - J_{s-1}$ and any $y_i^{s-1} < 0$, $(z^{s-1} - z_l)/|y_i^{s-1}| < c_j/a_{ij}$, no integer solution can be obtained for the partial solution at state $(s-1)$. This suggests it might be better to select a variable for which the above expression is not satisfied, for most of the negative $y_i^{s-1}$. This expression represents a relative measure of feasibility to optimality in augmenting the partial solution at state $(s-1)$. This means if we select a variable for which the above expression is satisfied for most of the negative $y_i^{s-1}$, this is more likely to lead to the infeasible solution away from the optimality. In the light of the above theorem, the following procedure is suggested.

1. Calculate for each variable $x_j$

$$t_j = -\infty \qquad j \in J_{s-1} \text{ and } x_j \text{ is integer in the linear programming solution}$$

$$= -\sum_{\substack{\text{all } y_i^s<0 \\ a_{ij}>0}} [c_j/a_{ij} - (z^s - z_l)/|y_i^s|] \tag{36}$$

2. Select $x_{j_s}$ for which

$$t_{j_s} = \max_{\text{all } j} \{t_j\}.$$

This procedure is arbitrary and hence does not necessarily lead to the best possible choice of $j_s$.

Once $x_{j_s}$ is selected, it can be fixed to any integer value in $\{l_{j_s}^{s-1} \le x_{j_s} \le u_{j_s}^{s-1}\}$. Before the algorithm backtracks to state $(s-1)$ all these values must be considered in a definite order. Let $x_{j_s}^*$ denote the value of $x_{j_s}$ in the linear programming solution. Krolak[6, 7] has used the following procedure to make a choice of a value at which $x_{j_s}$ is to fix. Pick either the next highest integer or the next lowest integer to $x_{j_s}^*$ as the first choice for $x_{j_s}$. Choose the other of these two values second. For succeeding choices alternate between the smallest value greater than $x_{j_s}^*$ not considered before and the largest value smaller than $x_{j_s}^*$ not considered before until all values between $l_{j_s}^{s-1}$ and $u_{j_s}^{s-1}$ are considered. The justification of doing so is as follows. Suppose $x_{j_s}$ is set to some value, say $\bar{x}_{j_s}$, such that $l_{j_s}^{s-1} \le \bar{x}_{j_s} \le x_{j_s}^*$ $(x_{j_s}^* \le \bar{x}_{j_s} \le u_{j_s}^{s-1})$. Consider the situation when one of the necessary conditions (given by Theorems 1 and 2) fails or the outcome of the resulting linear program at state $s$ falls in one of the first two categories. This immediately implies that for any value of $x_{j_s}$ between $l_{j_s}^{s-1}$ and $\bar{x}_{j_s}$ ($\bar{x}_{j_s}$ and $u_{j_s}^{s-1}$), no better integer solution can be obtained, since the objective function at state $s$ is a concave function of $x_{j_s}$.

This procedure somewhat increases the storage requirements on a computer, but the improved efficiency of the algorithm justifies its use.

## 7. STATEMENT OF THE ALGORITHM

A general outline of the algorithm for solving an all integer problem is presented here. Note that the imbedded heuristic search does not have any effect on the exactness of the algorithm. It acts like a catalyst, which helps the algorithm to converge faster. Also assume that the linear programming part of the algorithm is solved using the principle of decomposition.

*The algorithm*

The problem is to

maximize $z = cx$

subject to $Ax \le b,$

$x \ge 0$ and integer

*Step* 0: (0A) Find the upper and lower bounds on all variables.

(0B) If any integer solution is known, use this as a starting point and use the direct search in order to find a locally optimal solution.

(0C) Solve the above problem as a continuous linear program, using the principle of decomposition. Use the heuristic search at every iteration. Denote the linear programming solution by $z_u$ and the best known integer solution by $z_l$, if any. If no integer solution is known, set $z_l = 0$.

(0D) Initialize the matrix $\eta$ as empty.

(0E) Add the constraint $cx \ge (z_l + 1)$.

(0F) Make a choice of the variable to fix and the value at which to fix it and enter this information along with this variable's bounds in the first column of $\eta$. The algorithm is now in state 1, i.e., $s = 1$. Also initialize a variable $t = 1$, where $t$ corresponds to the column of $\eta$ in which the information for the variable $x_{j_s}$ is recorded.

*Step* 1: Find the conditional bounds on all free variables in $N - J_s$.

(1A) If for any free variable $x_j$, $l_j^{s+1} = u_j^{s+1}$. Enter this information in the 1st empty column, from the left, of $\eta$.

(1B) If $l_j^{s+1} > u_j^{s+1}$ for any free variable $x_j$, go to step 5.

Test the necessary condition specified by Theorem 2. If it fails, go to step 5.

*Step* 2: Solve the resulting linear program.

(2A) Infeasible solution. Go to step 5.

(2B) Integer optimal feasible solution. Replace the value of $z_l$ by the new value and make the corresponding change in the constraint $cx \ge (z_l + 1)$. Go to step 5.

(2C) Non-integer feasible optimal solution. Generate a surrogate constraint and add it to the original set of constraints. (Note: This constraint is not used while solving any linear programming problem.)

*Step* 3: Applying the heuristic search.

(3A) If a better solution is found, replace the value of $z_l$ and also make the appropriate change in the constraint $cx \geq (z_l + 1)$.

*Step* 4: $s = s + 1$. Make a choice of a free variable and fix it. Enter the appropriate information in $\eta$ and $J_s$. Go to Step 1 (forward step). Let $t$ be this column number.

*Step* 5: (Backward Step)

(5A) Cancel all columns to the right of column $t$ (i.e., set all elements in this column to zero). If $t = 1$ and $\eta(2, t) = \eta(3, t)$ terminate the algorithm.

(5B) For $t > 1$ the following cases are possible:

(a) if $\eta(R, t) \neq \eta(3, t)$, go to (5C)

(b) if $\eta(2, t) = \eta(3, t)$, go to (5D).

(5C) Four possible cases: If

(a) $\eta(4, t) \leq \eta(2, t) < \eta(3, t) \leq \eta(5, t)$,

1. set $\eta(2, t) = \eta(3, t)$
2. set $\eta(3, t) = \eta(3, t) + 1$, if $\eta(3, t) < \eta(5, t)$
   or $= \eta(5, t)$, if $\eta(3, t) = \eta(5, t)$
3. set $\eta(4, t) =$ new $\eta(3, t)$.

Go to Step 1.

(b) $\eta(4, t) \leq \eta(3, t) < \eta(2, t) \leq \eta(5, t)$

1. set $\eta(2, t) = \eta(3, t)$
2. set $\eta(3, t) = \eta(3, t) - 1$, if $\eta(3, t) > \eta(4, t)$
   or $= \eta(4, t)$, if $\eta(3, t) = \eta(4, t)$
3. set $\eta(5, t) =$ new $\eta(3, t)$.

Go to Step 1.

(c) $\eta(2, t) < \eta(3, t) = \eta(4, t) \leq \eta(5, t)$,

Go to (5D).

(d) $\eta(4, t) \leq \eta(5, t) = \eta(3, t) < \eta(2, t)$,

Go to (5D).

(5D) Find the first column from the right, say column $\bar{t}$, for which $\eta(2, \bar{t}) \neq \eta(3, \bar{t})$. Cancel all columns to the right of column $\bar{t}$. Let $h = \eta(2, \bar{t})$. Following cases are possible: If

(a) $\eta(4, \bar{t}) \leq \eta(2, \bar{t}) < \eta(3, \bar{t}) \leq \eta(5, \bar{t})$

1. set $\eta(2, \bar{t}) = \eta(3, \bar{t})$,
2. set $\eta(3, \bar{t}) = h - 1$, if $\eta(4, \bar{t}) \leq h - 1$,
   or if $\eta(4, \bar{t}) > h - 1$
   set $\eta(3, t) = \eta(3, \bar{t}) + 1$, if $\eta(3, \bar{t}) < \eta(5, t)$,
   or $= \eta(5, \bar{t})$, if $\eta(3, \bar{t}) = \eta(5, \bar{t})$.
   set $t = \bar{t}$. Go to Step 1.

(b) $\eta(4, \bar{t}) \leq \eta(3, \bar{t}) < \eta(2, \bar{t}) \leq \eta(5, \bar{t})$

1. set $\eta(2, \bar{t}) = \eta(3, \bar{t})$
2. set $\eta(3, \bar{t}) = h + 1$, if $\eta(5, \bar{t}) \geq h + 1$,
   or if $\eta(5, t) < h + 1$
   set $\eta(3, \bar{t}) = \eta(3, \bar{t}) - 1$, if $\eta(3, \bar{t}) > \eta(4, t)$,
   $= (4, \bar{t})$, if $\eta(3, \bar{t}) = \eta(4, \bar{t})$
   set $t = \bar{t}$. Go to Step 1.

(c) $\eta(2, \bar{t}) < \eta(3, \bar{t}) = \eta(4, \bar{t}) \leq \eta(5, \bar{t})$

1. set $\eta(2, \bar{t}) = \eta(3, \bar{t})$,
2. set $\eta(3, \bar{t}) = \eta(3, \bar{t}) + 1$, if $\eta(3, \bar{t}) < \eta(5, \bar{t})$
   or $= \eta(5, \bar{t})$, if $\eta(3, \bar{t}) = \eta(5, \bar{t})$
3. set $\eta(4, \bar{t}) =$ New $\eta(3, \bar{t})$.
   set $t = \bar{t}$. Go to Step 1.

(d) $\eta(4, \bar{t}) \leq \eta(5, \bar{t}) = (3, \bar{t}) < \eta(2, \bar{t})$

1. set $\eta(2, \bar{t}) = \eta(3, \bar{t})$
2. set $\eta(3, \bar{t}) = \eta(3, \bar{t}) - 1$, if $\eta(3, \bar{t}) > \eta(4, \bar{t})$
   or $= \eta(4, \bar{t})$, if $\eta(3, \bar{t}) = \eta(4, \bar{t})$.

3. set $\eta(5, \bar{t}) =$ new $\eta(3, \bar{t})$.
   set $t = \bar{t}$. Go to Step 1.

Step 3 can be used optionally. Step (0A), (0B), and (0C) can form a heuristic algorithm to solve an all integer linear programming problem.

## 8. EXAMPLE

In this section the algorithm is illustrated by solving the following problem:

$$\begin{aligned} \text{Maximize} \quad & z = x_1 + x_2 + x_3 \\ \text{subject to} \quad & x_1 + 2x_2 + 2x_3 + 2x_4 + 3x_5 \le 18 \\ & 2x_1 + x_2 + 2x_3 + 3x_4 + 2x_5 \le 15 \\ & x_1 - 6x_4 \le 0 \\ & x_2 - 8x_5 \le 0 \\ & \text{all } x_j \ge 0 \text{ and integer.} \end{aligned}$$

First this problem is solved using the heuristic and then it is also solved without using the heuristic.

*A. With Heuristic*

*Step* 0: (0A) $u' = (7,9,7,5,6)$ and
$l' = (0,0,0,0,0)$.

(0B) A solution $x^* = (0,0,0,0,0)$ is feasible.

Going through the direct search the solution obtained is
$x^* = (0,0,7,0,0)$ and $z^* = 7$.

(0C) Solve the problem as a linear program. The linear program solution is $x = (3.02, 5.59, 0.0, 0.50, 0.93)$ and $z = 8.61$.

(0D) $\eta$ is initialized as empty.

(0E) Add constraint
$x_1 + x_2 + x_3 \ge 8$.

(0F) Calculate $t_i$ for $i = 1$ to 5, $t_1 = 0.88775$, $t_2 = 0.88725$, $t_3 = -\infty$, $t_4 < 0$ and $t_5 < 0$.

Select $x_1$ to fix and let $x_1 = 3$.

$\eta =$

| |
|---|
| 1 |
| 3 |
| 4 |
| 0 |
| 7 |

*Step* 1: Conditional bounds.
$l^2 = (3,0,0,1,0)$
and $u^2 = (3,5,3,3,3)$

*Step* 2: Linear program.
(2A) infeasible solution.

*Step* 5: (Backward Step)

$\eta =$

| |
|---|
| 1 |
| 4 |
| 5 |
| 5 |
| 7 |

*Step* 1: Conditional bounds.
$l^2 = (4,0,0,1,0)$
and $u^2 = (4,4,2,2,3)$

*Step* 2: Linear program,
(2A) infeasible solution.

*Step* 5: Backward Step
Cancel this column. That terminates the algorithm.
*Solution is* $z = 7$ *and* $x = (0,0,7,0,0)$.
*Number of iterations* = 2.

*B. Without Heuristic*

*Step* 0: This is identical to one described above except at the end of step 0, we do not have any integer solution.

$s = 1$

$\eta =$

| |
|---|
| 1 |
| 3 |
| 4 |
| 0 |
| 7 |

*Step* 1: Conditional bound
$l^2 = (3,0,0,1,0)$
and $u^2 = (3,5,3,3,3)$

*Step* 2: Linear program
(2C) $z' = 7.5$, $x = (3,4.5,0,1,0.75)$
The surrogate constraint is
$1.5x_1 + x_2 + 1.5x_3 + 2.25x_4 \leq 11.25$

*Step* 4: $s = 2$. $x_2$ is selected and set equal to 4. (forward step)

$\eta =$

| | |
|---|---|
| 1 | 2 |
| 3 | 4 |
| 4 | 5 |
| 0 | 0 |
| 7 | 5 |

*Step* 1: Using surrogate constraint and original constraints.
$l^3 = (3,4,0,1,1)$
and $u^3 = (3,4,0,1,1)$

$\eta =$

| | | | | |
|---|---|---|---|---|
| 1 | 2 | 4 | 5 | 3 |
| 3 | 4 | 1 | 1 | 0 |
| 4 | 5 | 1 | 1 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 7 | 5 | 1 | 1 | 0 |

Solution is obtained.
$z^* = 7$ and $x = (3,4,0,1,1)$
Add constraint $x_1 + x_2 + x_3 \geq 8$.

*Step* 5: (Backward Step)

$\eta =$

| | |
|---|---|
| 1 | 2 |
| 3 | 5 |
| 4 | 5 |
| 0 | 5 |
| 7 | 5 |

*Step* 1: Conditional bounds
(1B) $l_4^3 = 1$ and $u_4^3 = 0$

*Step* 5: (Backward Step)

$\eta =$

| |
|---|
| 1 |
| 2 |
| 1 |
| 0 |
| 1 |

*Step* 1: Conditional bounds
$l^2 = (4,0,0,1,0)$
and $u^2 = (4,4,2,2,3)$
*Step* 2: Linear program
(2A) infeasible solution
*Step* 5: (Backward Step)

$\eta =$

| |
|---|
| 1 |
| 2 |
| 1 |
| 0 |
| 1 |

*Step* 1: Conditional bounds
$l^2 = (2,0,0,1,0)$
and $u^2 = (2,5,5,3,3)$
*Step* 2: Linear program
(2A) infeasible solution
*Step* 5: (Backward Step) Case 5.4A
Terminate the algorithm
*Solution is* $z = 7$ *and* $x = (3,4,0,1,1)$
Number of iterations = 5.

This illustrates how the imbedded heuristic helps the algorithm to converge faster. (Note that the two methods produced different optimal solutions. This implies this problem has multiple optimal solutions.)

## 9. COMPUTATIONAL RESULTS

This algorithm was tested by hand computation and a partial use of the computer. A computer program was written in FØRTRAN to solve a linear program using the principle of decomposition. For simplicity in coding this program, it was assumed that at least one feasible solution to the problem is known. In Step (0C) no heuristic is imbedded while performing the simplex iterations. The remaining steps in the algorithm are fully implemented without any discrimination.

Numerous test problems with up to 10 variables have been taken from the literature. The

Table 1. Computational experience for the all integer programming algorithm

| Problem designation | Problem size $n \times m$ | Number of Iterations | |
|---|---|---|---|
| | | No heuristic | With heuristic |
| Cook[13] | | | |
| 1 | 3×3 | 1 | 1 |
| 2 | 3×3 | 5 | 2 |
| 3 | 5×2 | 2 | 2 |
| 4 | 4×4 | 5 | 2 |
| 5 | 5×4 | 5 | 2 |
| 6 | 6×6 | 9 | 4 |
| 8 | 10×5 | 7 | 2 |
| Echols[11, 12] | | | |
| 2 | 3×3 | 2 | 2 |
| 3 | 6×4 | 11 | 4 |
| 4 | 5×2 | 4 | 2 |
| 5 | 6×3 | 1 | 1 |
| 10 | 5×2 | 8 | 4 |

number of iterations (execution of Step 1) for most of these problems is presented in Table 1. Each problem is solved twice, once skipping Steps 0B and 3. The columns corresponding to these are labeled "No Heuristic" and "With Heuristic".

The problems selected have been solved by other investigators, who are: Echols[11, 12] whose problems are randomly generated; and Cook[13], who had selected these problems from the literature and has documented them. Echols used some direct search techniques to find a heuristic answer to all the integer problem Cook solved his problems using a method for solving systems of linear diophantine inequalities.

The results summarized in Table 1 indicate that use of the heuristic dramatically reduces the number of required iterations.

## 9. CONCLUSIONS

The method described in this paper is a computationally feasible approach. Although no conclusive experience has been developed, the method contains some promising features which suggest its potential computational efficiency.

Although we have described the method in terms of the all integer programming problem, it is a straightforward matter to modify the method so that it can solve the general mixed integer programming problem[10].

The method can perhaps be improved if stronger tests for infeasibility can be developed. Other potential sources for improvement include the further exploitation of the use of decomposition, and the incorporation of cutting plane constraints into the algorithm.

## REFERENCES

1. A. M. Geoffrion, An Improved Implicit Enumeration Approach for Integer Programming, *Ops Res.* **17**, 437–454 (1969).
2. Egon Balas, An Additive Algorithm for Solving Linear Programs with Zero-One Variables, *Ops Res.* **13**, 517–546 (1965).
3. Egon Balas, Discrete Programming by the Filter Method, *Ops. Res.* **15**, 915–957 (1967).
4. A. M. Geoffrion, Integer Programming by Implicit Enumeration and Balas' Method, *SIAM Rev.* **7**, 178–190 (1967).
5. Fred Glover, A Multiphase-Dual Algorithm for the Zero-One Integer Programming Problem, *Ops Res.* **13**, 517–546, July–Aug. (1965).
6. Patrick D. Krolak, *The Bounded Variable Algorithm for Solving Integer Linear Programming Problems*, Doctoral Dissertation, Department of Applied Mathematics and Computer Science, Washington University, St. Louis, Mo. (1967).
7. Patrick D. Krolak, Computational Results of an Integer Programming Algorithm, *Ops Res.* **17**, 743–749 (1969).
8. C. E. Lemke and K. Spielberg, Direct Search Algorithms for Zero-One and Mixed Integer Programming, *Ops Res.* **15**, 892–914, Sept.–Oct. (1967).
9. A. Land and A. Doig, An Automatic Method of Solving Discrete Linear Programming Problems, *Econometrica*, **28**, 497–520, July (1960).
10. Anil B. Jambekar, *An Approach to Integer Programming*, Doctoral Dissertation, Washington University (1972).
11. R. E. Echols and L. Cooper, The Solution of Integer Linear Programming Problems by Direct Search, *J. ACM*, **15**, 75–84 (1968).
12. R. E. Echols, *The Solution of Integer Linear Programming by Direct Search*, M.S. Thesis, Washington University, St. Louis, Mo. (1966).
13. R. Cook, *An Algorithm for Integer Programming*, Doctoral Dissertation, Sever Institute of Technology, Washington University, St. Louis, Mo., January (1966).